Chuyi Liu and Jackson Mediavilla

A Gender Recognizer with OpenCV

Initial project idea and goals:

Our initial project idea was to develop a program that recognizes faces and determines the gender of the recognized faces. We aimed to have a fully functioning program with a graphical user interface as well.

Algorithm/Approach:

In this project we use C++/OpenCV as our language. OpenCV is a very powerful tool, just like Matlab. While Matlab may has advantages over OpenCV on algorithm and simulation, OpenCV has high computational efficiency and good for developing a real application. One thing we really appreciate about OpenCV rather than Matlab is it's highly compatible with different languages, and it's free for both academic and commercial use.

OpenCV provides three methods to recognize human faces: (1) EigenFace: call it with ctreaceEigenFaceRecognizer(); (2) FisherFace: call it with ctreaceFisherFaceRecognizer(); (3) LBPHFace: call it with ctreaceLBPHFaceRecognizer().

In our project, we provide all three methods but only use Local Binary Patterns(LBP) method for experiment, because of its high accuracy.

I read [1] and [2] to grab the basic idea of gender recognition algorithm. And following the introduction written by Philipp Wagner[4], LBP method compares the grayscale values of the eight neighbors around a certain pixel, that's why LBP face recognizer only works with grayscale images so we had to use imread(img,0) to load images. We take the grayscale value of the center pixel as a threshold, then when a neighbor has a lower grayscale value, we regard this pixel as 0, otherwise regard it as 1. Then we count LBP value for this center pixel. Shown as Fig.1.

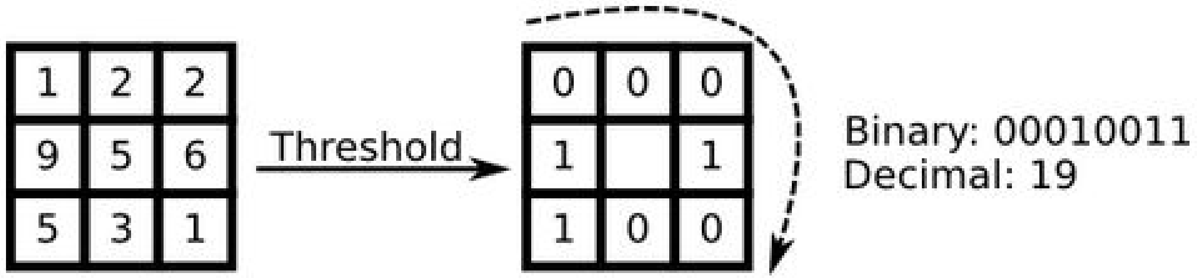


Fig.1 How to get LBP value of a pixel

By implementing this algorithm, we obtain an LBP value for each pixel in a given image. We would now be able to realize this algorithm in Matlab; however, we chose to use openCV for the convenience of using the createLBPHFaceRecognizer() function. In this stage, I found the tutorial of [3] is very helpful, though it's a program based on VisualStudio/Windows, it still very referential for our project.

As introduced in [6], in a LBP based face recognition algorithm, an image of a face is always divided into several regions(like 9 squares or more). Then the recognizer will extract a LBP histogram from each sub-region, as below:

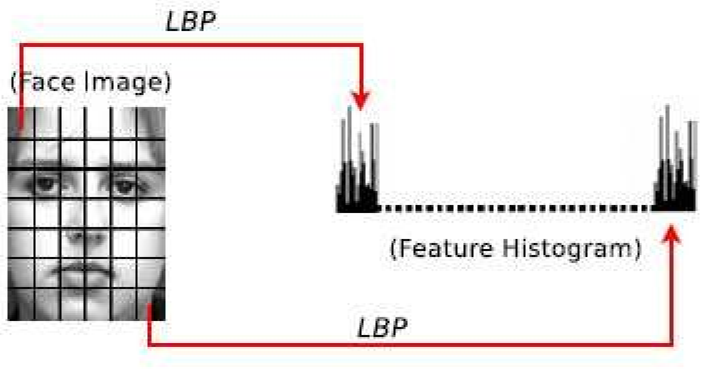


Fig.2 LBP histogram of a sub-region of an image

This extracted histogram describes the local texture and global shape of face images.

To maximize the general use of this program, we enable the user to input a .csv or .txt file to train a classifier with their own images and pick a test image as whatever they like, with a valid input command, of course.

In a linux terminal to realize that, the user just need to give a command as:

```
./gender  <csv file path>  <test image path>
```

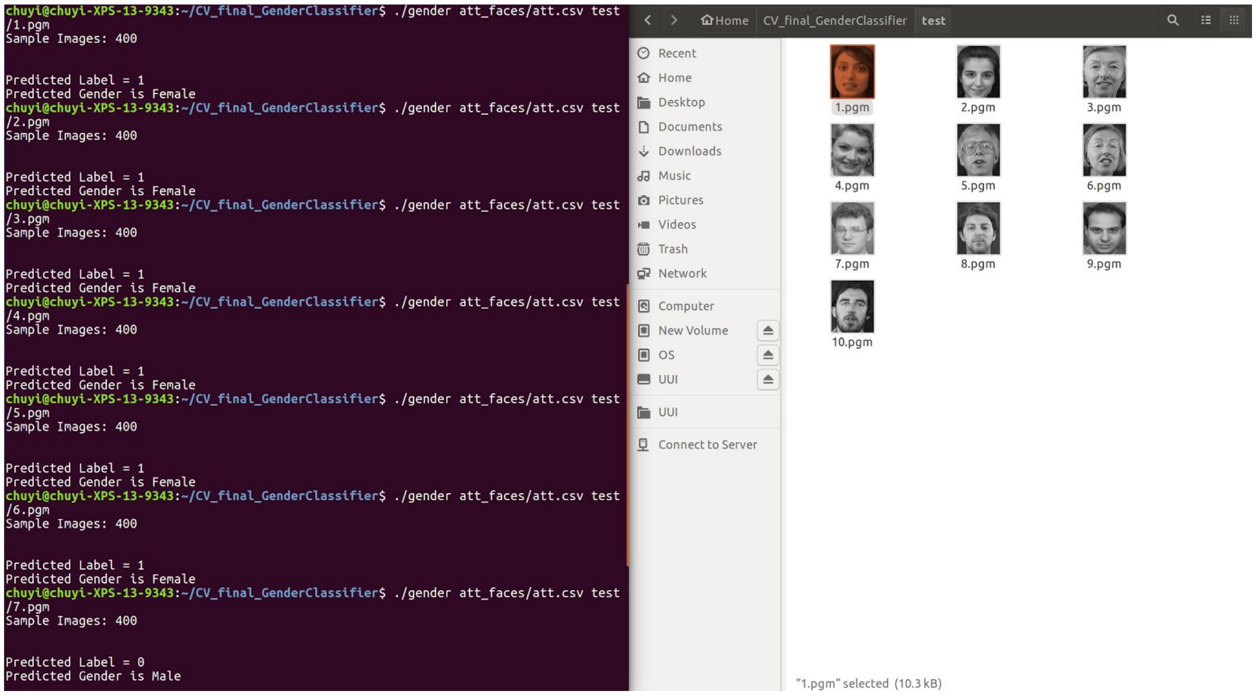
We considered two ways to get input images: (1) let the user to input an image file with command; (2) capture frames via a camera. In our experiments, we use (1) to get test images.

I want to give user more freedom to input images as they like, so I added a face detection part to detect faces in a given image automatically, then extract those faces as input for the gender classifier. It worked exactly, but sacrifice some accuracy though I didn't change a single step of the gender prediction stage. I doubt it's because the input is not ideal, so adding more sample images to train the classifier might solve this problem.

We wanted to use CAS-PEAL Face Database[7] to train the classifier but met some problems, it reported that the training images are not of the size, but we failed to figure out which image is incorrect, so finally we use ATT Face Dataset to train the classifier[5].

To allow the user to easily chose input images, we created a graphical user interface using QT. Upon execution, a main window pops up and requires that the user select an image file. For convenience, the user is brought straight to a directory containing the dataset of images. Once the user has selected an image file, the GUI executes our gender.cpp file to process the image.

Data Analysis and Results:



```
chuyi@chuyi-XPS-13-9343:~/CV_final_GenderClassifier$ ./gender att_faces/att.csv test
/1.pgm
Sample Images: 400


Predicted Label = 1
Predicted Gender is Female
chuyi@chuyi-XPS-13-9343:~/CV_final_GenderClassifier$ ./gender att_faces/att.csv test
/2.pgm
Sample Images: 400


Predicted Label = 1
Predicted Gender is Female
chuyi@chuyi-XPS-13-9343:~/CV_final_GenderClassifier$ ./gender att_faces/att.csv test
/3.pgm
Sample Images: 400


Predicted Label = 1
Predicted Gender is Female
chuyi@chuyi-XPS-13-9343:~/CV_final_GenderClassifier$ ./gender att_faces/att.csv test
/4.pgm
Sample Images: 400


Predicted Label = 1
Predicted Gender is Female
chuyi@chuyi-XPS-13-9343:~/CV_final_GenderClassifier$ ./gender att_faces/att.csv test
/5.pgm
Sample Images: 400


Predicted Label = 1
Predicted Gender is Female
chuyi@chuyi-XPS-13-9343:~/CV_final_GenderClassifier$ ./gender att_faces/att.csv test
/6.pgm
Sample Images: 400


Predicted Label = 1
Predicted Gender is Female
chuyi@chuyi-XPS-13-9343:~/CV_final_GenderClassifier$ ./gender att_faces/att.csv test
/7.pgm
Sample Images: 400


Predicted Label = 0
Predicted Gender is Male
```

We did a simple test for the gender classifier with 10 faces: 5 male faces and 5 female faces, it misrecognized 5.pgm, a male face, as a female face, but all other results were correct. So we assumed that the ideal accuracy of this classifier is ~90%, which might be lower if the input image is not very ideal.

For example, if we enable the *Detection Part - 2: an image of more than one people* Part in our source code, the accuracy dropped dramatically. Conclusively, this gender classification program achieved its fundamental goal, but to be a highly effective one that can be helpful for commercial use, it still need some improvement and more samples to train a better classifier.

Computer Vision Methods:

Using LBP features to recognize human faces.

Known Limitations and Possible Future Extensions:

ATT faces dataset contains 40 folders of human faces, but only 5 of them are female faces, which is a big drawback of this dataset for a gender classification purpose. In the future we can find some better dataset for this purpose and perhaps we can add age/expression/etc classifying functions by adding more labels to each training images in the .csv file.

Contributions:

Chuyi: I gathered several different face databases, then I picked CAS-PEAL and ATT face database as suitable for our purpose to train the gender classifier. Finally I realized the gender classification function with OpenCV/C++.

Jackson: Because we were initially interested in implementing a 3D human pose evaluator, I started my contributions to this project by doing a bunch of research on that. After reading the provided papers and doing some research into other people's projects, we decided that we would rather attempt the gender classification problem. We actually began implementation on the 3D human pose evaluator, but soon changed our minds. Once development on the gender classifier was started, I began writing the code for the GUI, which allows a user to select the image to process and then executes gender.cpp. Because I had never used QT, I relied heavily on the QT Reference Documentation.

Things We Learned:

Chuyi: How to apply an existing dataset to train a classifier for a certain purpose, like the ATT faces dataset we use to train this gender classifier, though it was made in order to realize an expression classifier, not a perfect one for a gender classifier, which still provide an acceptable accuracy. And I tried different ways for user to input images, and analyzed their influence on this program.

Jackson: As I mentioned above, we initially wanted to implement the 3D human pose evaluator project. Although we did not see this through, I learned a lot about different approaches to this problem including real-time human pose recognition in parts from a single depth image and the optical flow algorithm. From what we did implement, I learned about training a classifier to recognize gender. I also learned a lot about QT, C++, and OpenCV.

Advice for Future Students:

Chuyi: OpenCV FaceRecognizer class has some certain requirements on images to train or to test, for example they must be of the same size and you'd better load them as grayscale images(use cvtColor() function or just add a 0 in imread() function).

You might face some problems about dealing with the source images, but don't be as silly as me to waste a ton of time on that - try get another dataset.

Jackson: As it mentions in the handout, I would recommend choosing a project that enables you to set some high goals but also allows for a fallback strategy. I would also recommend choosing a project that already has a large dataset for training/testing. For the class in general, I recommend that students start their homeworks early. I feel like I learned a lot from the assignments after they were essentially finished and I was just working on making them better and faster. Allowing myself more time to do that for the assignments would have been very beneficial.

Reference:
1. Gender Classification with OpenCV
2. F. Samaria and A. Harter, "Parameterisation of a stochastic model for human face identification", 2nd IEEE Workshop on Applications of Computer Vision, December 1994, Sarasota (Florida).
3. C++开发人脸性别识别教程
4. Local Binary Patterns
5. ATT Faces Dataset
6. Face Detection using LBP features
7. CAS-PEAL Face Database